Exercice 1

The goal of this exercise is to manage student grades at an institution using a C++ Student class defined by: The following attributes:
• matricule: the student ID (auto incremented)
• name: name of a student
• nbrNotes: the number of notes of the student
• *tabNotes: table containing the notes of a student (dynamic allocation). The following methods:
• An initialization constructor
• A constructor with arguments
• A ~Student () destructor
• A Student copy constructor (const Student &)
• Getters and setters
• void input (): allowing the input of a student's grades
• void display(): allowing the display of a student's information
• float average(): returns as result the student's grade average.
• bool admitted (): returns as result the value true, if a student is admitted and the value false, otherwise. A student is considered admitted when the average of his marks is greater than or equal to 10.
• bool compare(): which compares the average of the two students, returns true if two students have same average and false in case the averages are different

Exercice 2

business layer

Define a class Room with the following attributes: id, code, and label.
Define accessors to the various attributes of the class.
Define a constructor to initialize attributes of a room object to values entered by the user.
Knowing that Id should be auto-increment.
Define the toString() method to display the current room information.

Data access layer

Create the generic interface IDao with the methods:
boolean create (T o): Method allowing to add an object o of type T.
boolean delete (T o): Method allowing to delete an object o of type T.
boolean update (T o): Method to modify an object o of type T.
T findById (int id): Method used to return an object whose id is passed as a parameter.
List<T> findAll(): Method to return the list of objects of type T.
Create the SalleService class which implements the IDao interface. In this class the data will be stored in a collection of type List.

Presentation layer
In a test class: Create five rooms. Display the list of rooms. Delete a room. Edit room information. Display the list of rooms.

Objectifs :

- Découvrir la programmation en couche
- Définir les propriétés et méthodes d'une classe
- Définir un constructeur d'initilialisation
- Créer une instance de classe
- Utiliser les variables de classe (static)
- Utiliser les interfaces génériques
- Utiliser les collections

Énoncé :

Couche métier

1. Définir une classe Salle avec les attributs suivants : *id, code et libelle.*
2. Définir les accesseurs aux différents attributs de la classe.
3. Définir un **constructeur** permettant d'initialiser les attributs d'un objet salle par des valeurs saisies par l'utilisateur. Sachant que Id doit être auto-incrément.
4. Définir la méthode **toString ( )** permettant d'afficher les informations de la salle en cours.

Couche accès aux données

1. Créer l'interface générique **IDao** avec les méthodes :
    - boolean create (T o) : Méthode permettant d'ajouter un objet o de type T.
    - boolean delete (T o) : Méthode permettant de supprimer un objet o de type T.
    - boolean update (T o) : Méthode permettant de modifier un objet o de type T.
    - T findById (int id) : Méthode permettant de renvoyer un objet dont id est passé en paramètre.
    - List <T> findAll ( ) : Méthode permettant de renvoyer la liste des objets de type T.
2. Créer la classe **SalleService** qui implémente l'interface **IDao**. Dans cette classe les données seront stockés dans une collection de type List.

Couche de présentation

1. Dans une classe de test :
    - Créer cinq salles.
    - Afficher la liste des salles.
    - Supprimer une salle.
    - Modifier les informations d'une salle.
    - Afficher la liste des salles.

    //Source : www.exelib.net